Chapter Six: Contents

(Selectors/Iteration Databases – LA-UR 00-1725)

Disclaimer

These archived, draft documents describe TRANSIMS, Version 1.1, covered by the university research license. However, note that the documentation may be incomplete in some areas because of the ongoing TRANSIMS development. More recent documentation (for example, Version 2.0) may provide additional updated descriptions for Version 1.1, but also covers code changes beyond Version 1.1.

Chapter Six: Figures

Chapter Six—Selectors/Iteration Databases

1. INTRODUCTION

1.1 Overview

A key distinguishing feature of TRANSIMS is the process known as iterative feedback. Feedback provides a natural way to tailor models (of activity locations, mode selections, route planning, etc.) to specific, possibly overlapping, subpopulations. Feedback enables the overall computational system to reflect "learned" behavior within the simulated population represented. Feedback involves two crucial processes:

- Biased selection – defining a subpopulation based on any static or dynamic information about travelers available to TRANSIMS.
- Updating travelers – revising the selected subpopulation's use of the transportation system by controlling the quality of information about the system available to them.

The information about travelers available to TRANSIMS consists of the traveler-specific data contained in population, activity, plan, vehicle, and simulation output files. This data is all generated by TRANSIMS under specific hypotheses about the transportation network. By carefully controlling the hypotheses, TRANSIMS can be used to steer travelers toward certain choices.

The mechanics of controlling information flow among TRANSIMS modules is discussed in the Input/Output section of each module's description. This chapter describes the Selector/Iteration Database and how it works together with an iteration script to control the overall TRANSIMS framework. A typical TRANSIMS study involves repeated iteration between modules. There is no single, "standard" iteration script because different study designs involve different iteration schemes.

One important example of feedback is in solving the traffic assignment problem. The simplest version of this uses a loop between the Route Planner and Traffic Microsimulator modules. On the first pass of the Route Planner, routes are chosen under the hypothesis that travel time is well represented by free speeds on the network (i.e., that travelers do not interact). Correction for traveler interactions can be applied simply by making available to the Route Planner information about actual travel times produced by the Traffic Microsimulator[1]. With this information, the Route Planner will choose different routes for most travelers, resulting in different travel times. In this case, updating travelers is accomplished by re-running the Route Planner with an updated travel time table. However, there is still a wide range of different feedback schemes for this problem which depend on the selection step – exactly which travelers are to be run through the Route Planner with updated travel time information. One selection process is to choose a certain fraction of travelers uniformly at random. The Selector/Iteration

[1] Notice that there is no requirement to provide *correct* travel time information – it might be noisy, or averaged together with travel times used in the previous run.

Database described below supports much more sophisticated processes, though. For example, one could select only travelers with automobile drives of an hour or more who cross a geographic feature (like a river).

Of course, there are many more information flows in TRANSIMS than just the travel time table. Every TRANSIMS module can be used to update only a selected subpopulation using information provided by the framework. In effect, this is like providing a separate model for every conceivable subdivision of the population without the need for fitting each model separately. For example, work location is chosen using a single simple model for the entire population. If people who commute by bus across a river are assigned work locations poorly, selecting that subpopulation and running the work location assignment model with slightly different input information can change the poorly selected locations for that subpopulation with no change in the model itself.

Notice that a single traveler might be in *two* subpopulations – for example, the previous subpopulation and the subpopulation assigned to households larger than five people who also have longer than average commutes—but no sophisticated correlation structure needs to be built into the model to handle such cases correctly.

Selection is based on both absolute criteria such as traveler's mode and on relative criteria such as the duration of a trip compared to the duration of all other trips in the subpopulation picked out by the absolute criteria. The relative criteria act as user-specified cost functions. Thus, we might select the 10% of travelers meeting some absolute criteria who have the longest actual travel time compared with their expected travel time.

Fig. 1 gives an indication of how the selection process works. Here data is collected on the travelers' incomes, their travel modes, the length of trips, whether they cross the river, and the relative length of the trip. All travelers with some collection of these characteristics, for example those on bus trips with income >$40k, are collected and the distribution of relative trip duration is formed. A portion of these travelers with the largest duration is selected to travel by a different mode.

Selection and Feedback

The iteration database:

Traveler	Income	Mode	>1 hour?	Cross river?	Relative duration	. . .
291362	$25K	bus	no	yes	1.2	. . .
291363	$34K	car	yes	no	1.6	. . .
291364	$42K	car	no	yes	1.1	. . .
291365	$ 0K	walk	no	no	1.0	. . .
291366	$38K	car	yes	yes	2.3	. . .
291367	$45K	bus	yes	no	1.4	. . .
291368	$30K	car	yes	yes	1.3	. . .

Selection criterion:	Selects travelers:
bus trips with income >$40K	291367
short trips crossing the river	291362 291364
long car trips not crossing the river, relative duration > 1.3	291363

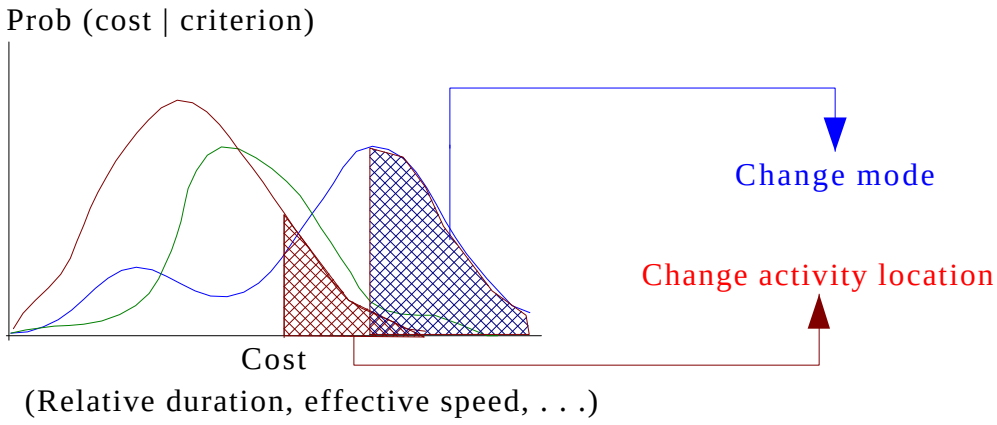

Fig. 1. The selection process.

1.2 The Iterative Process

Users can prepare an iteration script to control the entire iteration process. The script uses special control commands specifically developed for this iteration of TRANSIMS components. It enables the user to filter results, run repeated iterations, establish stopping criteria, and perform a host of other operations that make the analyst's job less labor intensive.

During each iteration, the iteration script controlling the current study typically invokes a Selector/Iteration Database. (The script might even use a different Selector/Iteration Database for each iteration in a study.) When a Selector/Iteration Database runs, it usually will do the following:

- Read information about the travelers from the Selector/Iteration Database.
- Examine each traveler and decide whether to
  - regenerate his or her activities using the activity generator,
  - select a new route between his or her existing activities using the Route Planner, or
  - retain his or her existing activities and the planned route between them.
- Write the selections made for each traveler into data files that can be read by the Activity Generator and the Route Planner when they are executed.
- Summarize the selections made and the current state of the system into a Selector/Iteration Database statistics data file.

Fig. 2 and Fig. 3 illustrate a Selector/Iteration Database's decision-making process.

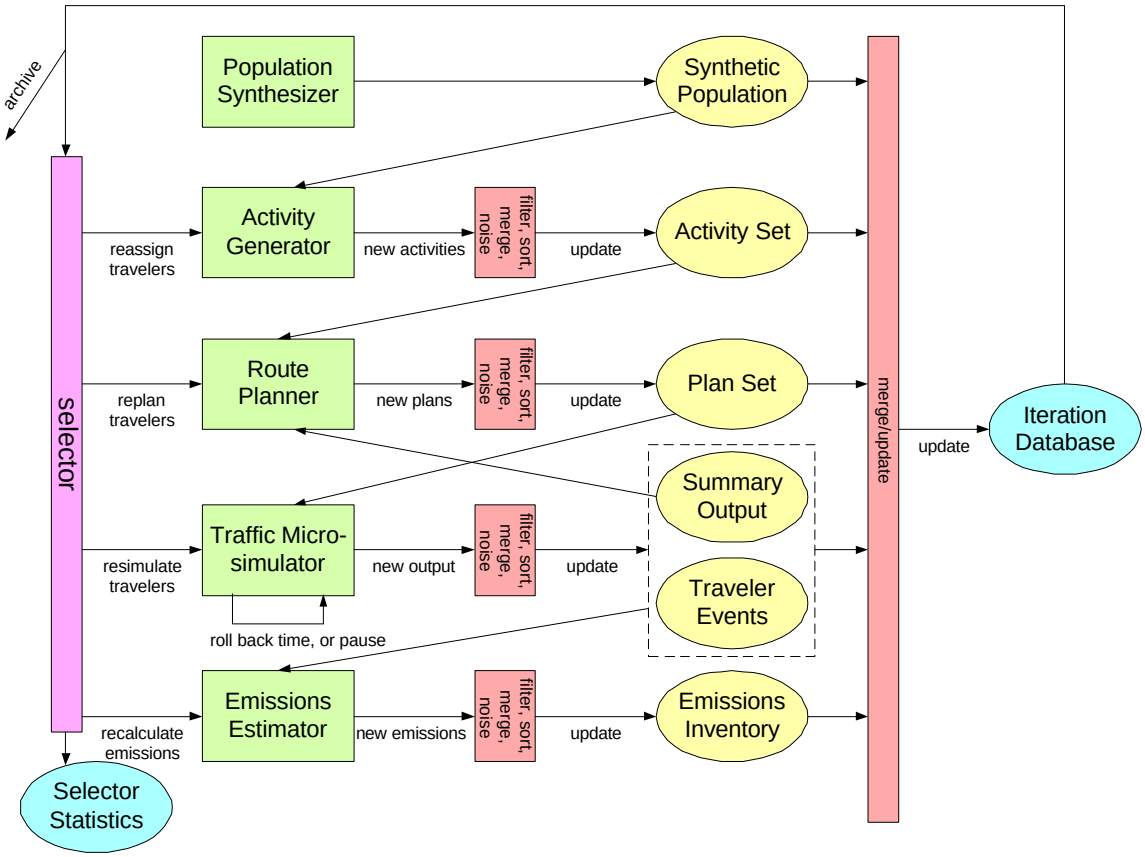

Fig. 2. Location of the Selector/Iteration Database within a typical TRANSIMS experimental design.

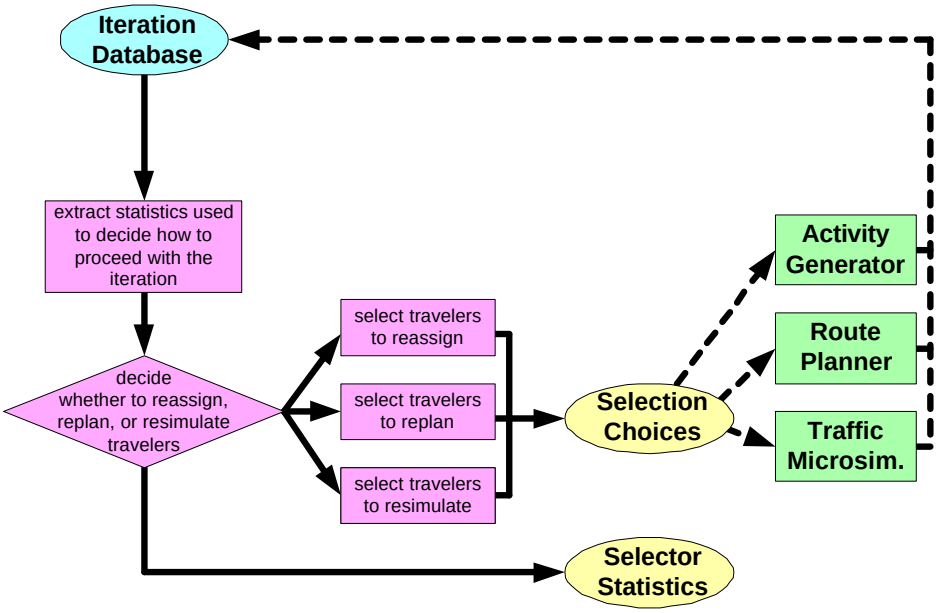

Fig. 3. Typical Selector/Iteration Database logic.

After the Selector/Iteration Database completes the selection process for all travelers, the Activity Generator, Route Planner, or Traffic Microsimulator runs to calculate the updated activity set, plan set, or microsimulation output files, respectively (according to the decisions made by the Selector/Iteration Database).

The iteration script will re-invoke a Selector/Iteration Database again at the start of the next iteration in the study. Fig. 4 shows examples of four possible progressions, as determined by the Selector/Iteration Database.

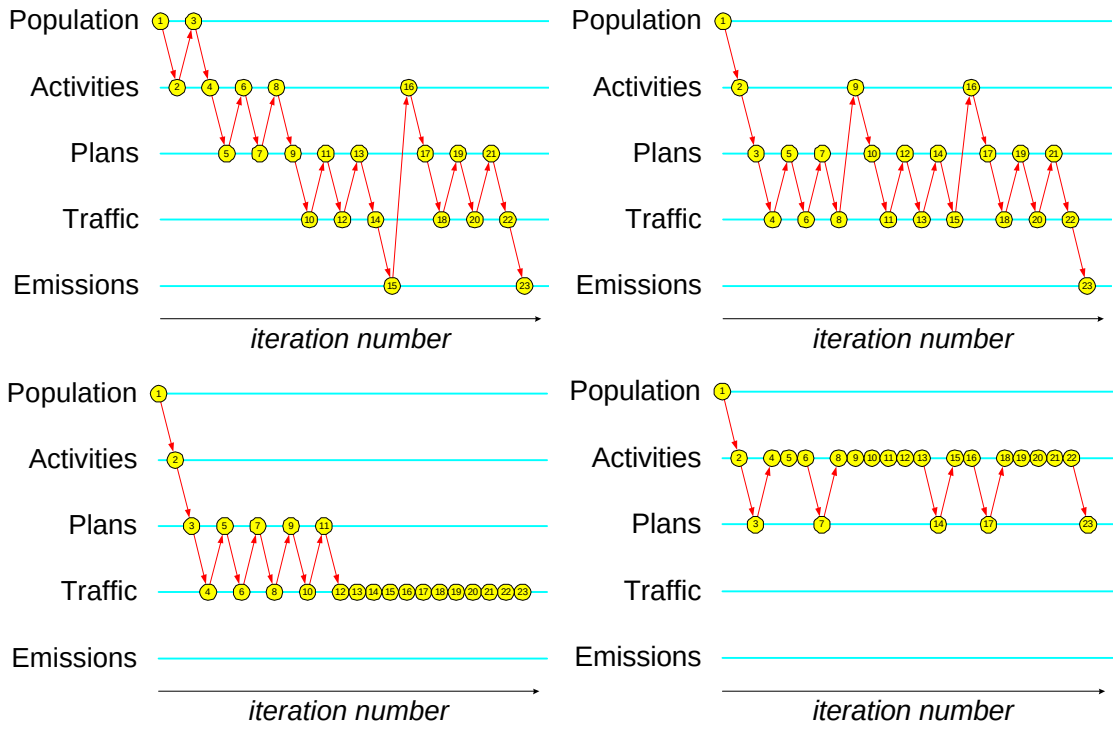

Fig. 4. Four examples of iteration progressions.

The Selector/Iteration Database is the archive of information about travelers across iterations. The Selector/Iteration Database uses this information to make its selection decisions. The data contained in the database are chosen by the user from:

- The fields of the population, activity, and plan files—for example, income, mode preference, or the expected duration of a trip.
- Information extracted from detailed Traffic Microsimulator event output—for example, the actual duration of a trip.
- Information deduced from combinations of the previous two—for example the duration of a trip relative to its expected duration.

The left side of Fig. 5 shows this data flow into the Selector/Iteration Database.

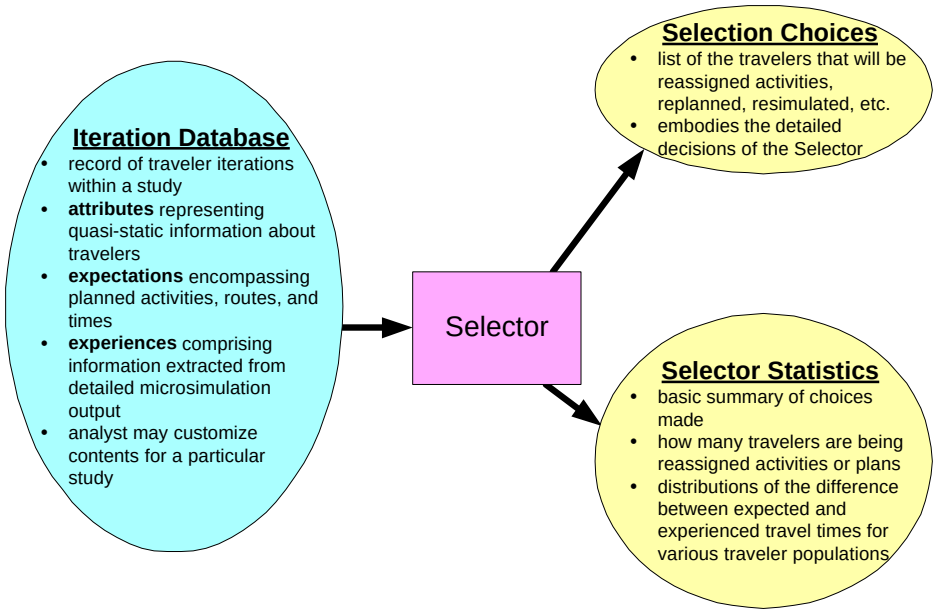

Fig. 5. Typical Selector/Iteration Database data flow.

The Selector/Iteration Database also has two principal outputs: Selector/Iteration Database statistics and selection choices. The selection choice files simply list the travelers that will be reassigned activities or will be re-planned, re-simulated, etc. These files embody the Selector/Iteration Database's detailed decisions.

Selector/Iteration Database statistics provide a basic summary of the choices a Selector/Iteration Database makes (e.g., how many travelers are being re-planned, distributions of the difference between expected travel times, and experienced travel times for various traveler populations). The right side of Fig. 5 shows this data flow out from the Selector/Iteration Database.

1.3 TRANSIMS Framework Flexibility

The framework's flexibility allows for countless variations in the iteration process. For example, in some studies, part of the Selector/Iteration Database may run again after the Activity Generator or Route Planner completes its execution. Thus, the Selector/Iteration Database can decide which of the generated activities or plans will be accepted for travelers. Those not accepted are discarded and new activities or plans are produced.

Users can also design Selectors/Iteration Databases that will feed travelers to the Activity Generator or Route Planner one-by-one so that the Selector/Iteration Database, Activity Generator, Route Planner, and Traffic Microsimulator all execute simultaneously, with their coordination controlled by the Selector/Iteration Database. This may increase the computational efficiency of a study, thus allowing for new experimental designs with finely controlled iteration.

The iteration script has the potential to make additional choices, such as the following:

- which version of the Activity Generator, Route Planner, or Traffic Microsimulator will run during the present iteration.
- if transit schedules will be adjusted or vehicles added or removed from the transit fleet.
- if network characteristics (such as traffic signal timing, congestion pricing, or roadway information signs) will be altered.
- which travelers receive data from traffic information systems.
- whether to complete the study (i.e., end the iteration) because the iterations have converged sufficiently (or diverged).

Several Selector/Database Iteration implementations have been written that have use in typical transportation planning studies. For example, Fig. 3 shows a typical iteration scheme that is set up by the Selector/Iteration Database script. In this scheme, activities, plans, and microsimulations are iterated until traffic behavior on the network stabilizes. It is not difficult for analysts to write additional Selectors/Iteration Databases for their own specialized studies.

2. ALGORITHM

2.1 Overview

This section specifies the intended interface to a suite of tools providing Selector/Iteration Database functionality. These tools are provisionally known as the Collator, Stratifier, and Selector/Iteration Database. **Although these tools are not available in this distribution of TRANSIMS**, their interface and design is documented here because it is a part of the final TRANSIMS package.

Each different module of TRANSIMS provides information that can be used to fill in some of the fields. For example, after running the Population Synthesizer, demographic information can be collected; after running the Route Planner, expected travel times can be collected. The Collator can be run after each module and will fill in all the fields in the ITDB that depend on that module with the most recent data available.

As its name indicates, the Collator's main function is to gather data from disparate sources (e.g., activity files, plan files, event files) into a single table keyed by traveler ID. In addition, the Collator will provide some commonly used processing algorithms described below.

2.2 Processing Algorithms

One form of processing determines whether a traveler passes through a particular geographic region or whether the traveler crosses from one region to another. This processing is triggered if the configuration file contains the `ITDB_USE_TRAVERSE` or `ITDB_USE_CROSS_BOUNDARY` keys.

Both keys' values should be a semicolon-separated list of user analysis zones (UAZ), which are defined by the user. UAZs may be a coarsening of traffic analysis zones (TAZ), but they may also be completely different. The UAZs will be placed in separate files of the form to be used to specify regions for the Output Visualizer. That is, the first line gives the number of polygons to follow. Each line after that gives a polygon (UAZ) ID followed by the number of vertices in the polygon, then by the x,y,z values for each vertex. The interpretation of UAZs will ignore the "z" value.

The coordinate system will be the one used by the network node table. There could be more than one set of UAZs: one might be used to define the Central Business District, for example, while another distinguishes the two sides of a river. There will be a sequentially numbered set of UAZs, one for each `UAZ_FILE_n` key. The first value of *n* must be "1."

Another form of processing that the Collator will do is to calculate the following simple functions of pairs of columns:

```
DIFF (A, B) = A - B
REL_DIFF (A, B) = (A - B) / B
RATIO (A, B) = A / B
```

These can be specified using configuration file keys of the form `ITDB_USE_DIFF <A>, <B>; <C>, <D>` , where `<A>`, `<B>`, etc., are the names of fields in the ITDB. (`REL_DIFF` could be calculated using DIFF and RATIO, but it is such a common operation that TRANSIMS provides a simple means of doing it.)

2.2.1 Stratifier

The Stratifier uses a combination of built-in algorithms on the information contained in the ITDB to stratify or classify trips. The classification is stored in the ITDB as indexes into a set of n-way user-specified tables. The tables themselves can be reconstructed from the configuration file used with the Stratifier.

2.2.2 Selector/Iteration Database

A Selector/Iteration Database uses the ITDB to select a set of travelers. Each algorithm can be coded as a separate executable, or could be incorporated into a single monolithic Selector/Iteration Database executable. A Selector/Iteration Database algorithm has a name, a goal, a cost function (optionally), and optional parameters associated with it. These attributes are defined by configuration file keys `SEL_ALGORITHM_n`, `SEL_GOAL_n`, `SEL_COST_n`, and `SEL_ALGORITHM_n_PARAMETER_m`, respectively, where *n* and *m* are replaced by positive integers. The name is user defined, but must be unique across algorithms. The goal is either to re-route the traveler or to make use of one of the feedback pathways defined in the activity generator. Currently, these are as follows:

A = full activity generation by matching to survey households.

B = New location given updated travel times

C = Change mode to given mode string

D = New location given new mode string and updated travel times

E = Change activity times

The cost function must be the name of a field present in the ITDB when the Selector/Iteration Database is run. Each algorithm may require its own set of parameters. Because we cannot know ahead of time what all of the parameters will be, we refer to them in the configuration file by number. The designer of a Selector/Iteration Database algorithm must ensure that the parameters are used correctly. No Selector/Iteration Database algorithm is required to handle every goal.

The required fields in the ITDB are as follows:

- `TRAVID`
- `HOUSEID`
- `TRIPID`

The user may specify the following additional fields of raw data, all of which come from either the population, activity, plan, or event files, using keys of the form `ITDB_USE_<field_name>` , where `<field_name>` is one of

- `DESIRED_ARRIVAL_TIME` (this implies the fields `DESIRED_ARRIVAL_TIME_UB`, `DESIRED_ARRIVAL_TIME_A`, `DESIRED_ARRIVAL_TIME_B`, which can also be specified separately)
- `EXPECTED_ARRIVAL_TIME`
- `ACTUAL_ARRIVAL_TIME`
- `NUM_STOPS`
- `TIME_STOPPED`
- `TOTAL_DISTANCE`
- `TOTAL_TIME`
- `MODE_PREF`
- `EXPECTED_DURATION`
- `HOUSEHOLD_DEMOS`
- `PERSON_DEMOS`

The user may specify the presence of the following additional fields of processed data using associated configuration file keys:

- `GEOM_DISTANCE`
- `TRAVERSE_UAZ<m>_<n>`
- `CROSS_BOUNDARY_UAZ<m>`
- `DIFF_<c1>_<c2>`
- `REL_DIFF_<c1>_<c2>`
- `RATIO_<c1>_<c2>`

In the above, *m*, *n*, and *p* are integers, and *c1* and *c2* are the names of fields of raw data. There will be a single value indicating that the calculation did not return a number (to handle, divide by zero or missing data).

2.3 Building Fields

The Stratifier builds fields named `STRAT_TABLE_n` using data in the previous fields. The value of this field is the index into a cell in an n-way table. The details of how binning is accomplished are determined by

- configuration file keys specifying the number of variables in the table (or levels in the tree),
- the cost function used for the split at each level (which is the name of a field in the ITDB),
- the number of bins (or degree of the nodes at that level of the tree), and
- optionally, the boundaries of the bins.

Bin boundaries do not need to be specified for categorical data. If they are not specified for ordinal data, quantiles will be used. The example below is an annotated portion of a configuration file using the keys described above.

```
# Specify files containing two sets of user analysis zones.
ITDB_UAZ_1  /tmp/zones_1
ITDB_UAZ_2  /tmp/zones_2

# Add two boolean fields to the ITDB: one is true if a traveler
# passes through zone 2 in the first set of UAZs; another is
# true if a traveler passes through zone 4 in the second set of UAZs.
# These fields will be named TRAVERSE_UAZ1_2 and TRAVERSE_UAZ2_4.
ITDB_USE_TRAVERSE  UAZ1=2; UAZ2=4

# Add three boolean fields to the ITDB: the first is true if a
# traveler's origin and destination are in different zones as defined
# by the third set of UAZ's; the second is true if they are in
# different zones as defined by the second set of UAZ's; the third uses
# the fourth set of UAZ's. They will be named CROSS_BOUNDARY_UAZ3,
# CROSS_BOUNDARY_UAZ2, and CROSS_BOUNDARY_UAZ4.
ITDB_USE_CROSS_BOUNDARY   UAZ3 ; UAZ2; UAZ4

# These keys make sure the relative difference between
# actual and expected trip duration are in the ITDB for
# later use in a cost function.
ITDB_USE_EXPECTED_DURATION  1
ITDB_USE_TOTAL_TIME  1
ITDB_USE_REL_DIFF EXPECTED_DURATION; TOTAL_TIME

# The first stratification will be based on three splits.
STRAT_LEVELS_0 3

# The first split in the first stratification will be based on
# whether a traveler passed through zone 2 in the first set of UAZs.
# Note that this is categorical data - there will be one bin
# for each of the two categories.
STRAT_SPLIT_0_0 TRAVERSE_UAZ1_2
```

```
# The second split in the first stratification will be on trip length.
# DISTANCE is the name of a field in the ITDB.
STRAT_SPLIT_1_0 DISTANCE

# DISTANCE is a numerical field - we must specify how many bins.
STRAT_SPLIT_BINS_1_0 3

# We specify the bin boundaries for this split.
STRAT_SPLIT_VAL_0_1_0 10.0
STRAT_SPLIT_VAL_1_1_0 50.0

# The 3rd and final split in the first stratification will be on time.
# TIME is the name of a field in the ITDB. We will make four bins.
# We don't specify the bin boundaries - the code will use p-tiles for
# p bins.
STRAT_SPLIT_2_0 TIME
STRAT_SPLIT_BINS_2_0 2

When the stratifier runs, it will assign values from
# 0 to 2 * 3 * 4 -1 = 23 to each traveler in the column STRAT_0.
# Notice that it doesn't assume anything about the cost function to be
# used in the Selector/Iteration Database, it just partitions the
# travelers. There will also be a tool that reads the configuration
# file, takes a bin id or traveler id and says something like:

# TRAVERSE_UAZ1_2 true
# DISTANCE [10.0, 50.0]
# TIME     [73.4, 228.6]

# A similar tool could write out the occupancies of each cell in the
# table.

# Now the configuration file keys for the Selector/Iteration Database
# can refer to algorithms and stratifications:

# Each algorithm selects a set of travelers for some purpose.
# The SEL_GOAL key determines the purpose.
# This selection will be used to pick a new mode for some travelers.
# Other choices here are WORK_LOCATION, TIME,
SEL_GOAL_0  MODE.

# Each algorithm has a name associated with it.
# Use Chris's algorithm for selecting travelers.
SEL_ALGORITHM_0 CHRIS

# An algorithm may refer to a cost function that has
# been calculated by the stratifier or collator for each traveler
# and each trip.

# Use (expected - actual time)/(expected time) as the cost function for
# selecting.
# REL_DIFF_EXPECTED_DURATION_TOTAL_TIME is a field in the ITDB if
#     ITDB_USE_EXPECTED_DURATION  1
#     ITDB_USE_TOTAL_TIME  1
#     ITDB_USE_REL_DIFF EXPECTED_DURATION; TOTAL_TIME
# are all in the configuration file.
SEL_COST_0   REL_DIFF_EXPECTED_DURATION_TOTAL_TIME
```

```
# Most algorithms will be parameterized. If we knew ahead of time what
# all the algorithms were and what parameters they required, I would
# add them as configuration file keys. Instead, I will fake it,
# allowing for keys of the form:
SEL_ALGORITHM_1_PARAMETER_0 0.25

# To select 10% of  travelers at random for replanning, you might say
SEL_ALGORITHM_1 RANDOM SEL_GOAL_1 PLAN SEL_ALGORITHM_1_PARAMETER_0  0.10
```

3. AN EXAMPLE SELECTOR/ITERATION DATABASE AND SCRIPT

A simple Selector/Iteration Database is provided with this distribution of TRANSIMS. It can be used to update the iteration database after the Traffic Microsimulator is run. It can also select travelers at random or based on where they appear in each of several distributions of statistics. This Selector/Iteration Database is used in the script controlling iterations in the "multimode" scenario included in this distribution. The iteration script ties the iteration process together by running the components, merging files, and archiving output data. For further information on this script, please see the multimode documentation in Volume Five (*Software: Interface Functions and Data Structures*).

The Selector/Iteration Database's merge-update function is used to update the contents of the iteration database by using the latest population, activity, plan, and traveler event files generated by the Population Synthesizer, Activity Generator, Route Planner, and Traffic Microsimulator, respectively. This function is invoked if the `SEL_FILL_ITDB` configuration file key is true when the Selector/Iteration Database is run.

Appendix A summarizes the contents of the iteration database used by the Selector/Iteration Database described here. After the iteration database has been completed, the Selector/Iteration Database may be used to build a variety of cost functions, depending on the value of several configuration file keys.

In addition, the Selector/Iteration Database can generate changes to apply to sets of travelers using several hard-coded algorithms. These are described below, but not used by the Selector/Iteration Database script in calnet. Instead, it uses UNIX tools to process the (ASCII text) iteration database and generate its own feedback. This demonstrates the flexibility of the feedback and iteration process.

3.1 Hard-coded Algorithms

3.1.1 Duration Cost

$$c_{duration}(i_{traveler,leg}) = \frac{T_{actual} - T_{expected}}{T_{expected}}$$

where T_{actual} is the actual travel time for the trip as realized by the Traffic Microsimulator, and $T_{expected}$ is the expected travel time for the leg as estimated by the Route Planner. This measures travel time “frustration.”

3.1.2 Distance Cost

$$c_{distance}(i_{traveler,leg}) = \frac{D_{actual} - D_{geometric}}{D_{geometric}}$$

where D_{actual} is the actual distance traveled in the Traffic Microsimulator, and $D_{geometric}$ is the point-to-point Euclidean distance between the leg's endpoints. This measures how far out of his or her way the traveler goes.

3.1.3 Stopped Cost

$$\textit{Stopped cost: } c_{stopped}(i_{traveler,leg}) = \frac{T_{stopped}}{T_{actual}}$$

where $T_{stopped}$ is the time stopped in traffic, and T_{actual} is the total travel time. This measures the fraction of the time a traveler spends waiting.

3.1.4 Late Cost

$$c_{late}(i_{traveler,leg}) = A_{desired} - A_{actual}$$

where $A_{desired}$ is the arrival time desired by the Activity Generator, and A_{actual} is the actual arrival time realized by the Traffic Microsimulator. This measures how late the traveler is for his or her activity.

3.1.5 Effective Speed Cost

$$c_{speed}(i_{traveler,leg}) = \frac{D_{geometric}}{T_{actual}},$$

where $D_{geometric}$ is the point-to-point Euclidean distance between the leg's endpoints, and T_{actual} is the total travel time. This measures the traveler's effective speed through the network.

3.2 Traveler Actions

Once the cost functions have been built, they are sampled to determine what actions the travelers will take. The following steps are used to make these determinations:

Step One Select the fraction $f_{reassign}$ of the travelers uniformly at random, $c_{distance}(i_{traveler,leg})$, for reassignment of activity locations by the activity generator. Once this is done, use the Route Planner to reroute the travelers.

Step Two Select the fraction f_{remode} of the travelers with the highest effective speed costs, $c_{speed}(i_{traveler,leg})$, for reassignment of mode preferences by the activity generator. Once this is done, use the Route Planner to reroute the travelers.

Step Three Select the fraction f_{retime} of travelers with the highest late costs, $c_{late}(i_{traveler,leg})$, for reassignment of activity times by the Activity Generator. Once this is done, use the Route Planner to reroute the travelers.

Step Four Select the fraction $f_{reroute}$ of travelers with the highest duration costs, $c_{duration}(i_{traveler,leg})$, for rerouting by the Route Planner.

The end result of the selection choices consists of a pair of files containing a list of travelers whose activities must be regenerated (along with what type of regeneration must be done) and a list of travelers whose routes must be re-planned.

Appendix A: Description of Selector/Iteration Database Fields

Field Description	Description	Data Source
TRAVID	traveler's ID	population file
HOUSEID	the ID of the traveler's household	population file
TRIPID	the ID of the traveler's trip	plan file
LEGID	the ID of the trip's leg	plan file
DESIRED_ARRIVAL_TIME	the desired arrival time (measured in fractional hours) at the activity	activity file
DESIRED_ARRIVAL_TIME_UB	the upper bound for desired arrival time (measured in fractional hours) at the activity	activity file
DESIRED_ARRIVAL_TIME_A	beta-distribution parameter specified in the activity file	activity file
DESIRED_ARRIVAL_TIME_B	beta-distribution parameter specified in the activity file	activity file
EXPECTED_ARRIVAL_TIME	the expected arrival time (measured in seconds past midnight) at the activity	plan file
ACTUAL_ARRIVAL_TIME	the actual arrival time (measured in seconds past midnight) at the activity	traveler event file
NUM_STOPS	the number of stops signs the traveler encountered on this leg	traveler event file
TIME_STOPPED	the number of seconds the traveler was stopped in traffic on this leg	traveler event file
TOTAL_DISTANCE	the total distance (measured in meters) traveled on this leg	traveler event file
TOTAL_TIME	the total time (measured in seconds) traveled on this leg	traveler event file
GEOM_DISTANCE	the straight-line (Euclidean) distance (measured in meters) planned for travel on this leg	plan file
MODE_PREF	the traveler's mode preference for this leg (same integers as in activity file)	activity file
EXPECTED_DURATION	the expected duration (measured in seconds) of the current leg	plan file

Chapter Six: Index